

Funciones y ciclos (Parte II)

Document Object Model (DOM)

Competencias

- Reconocer los principales elementos del DOM para utilizar adecuadamente los componentes y métodos con que se accede y manipula un documento HTML.
- Ejecutar funciones creadas de JavaScript para interactuar con los elementos del DOM y manipular su comportamiento.

Introducción

Actualmente tenemos nociones de JavaScript y HTML, por lo que la forma de conectarlos es a través del Document Object Model o DOM. El DOM representa un documento HTML mediante una forma de árbol, que es posible manipular por nosotros. En este capítulo, veremos qué es el DOM y cómo podemos hacer uso de él para tener un control claro de los elementos que componen una página web.

Manejar estos conceptos te dará las bases para entender más adelante cómo capturar, manipular y utilizar eventos en JavaScript, permitiendo la interacción entre las aplicaciones y los usuarios, una de las características más relevantes de este lenguaje.

El concepto DOM

El DOM es un estándar adoptado por los navegadores web, creado por el W3C (*World Wide Web Consortium*), con el fin de ser un medio que permita a los programas y scripts, acceder y gestionar el contenido.

Para aplicar el DOM en el HTML mediante el uso de JavaScript, es necesario utilizar ciertas instrucciones muy específicas del propio lenguaje de programación, las cuales permitirán captar elementos del documento HTML como lo son: id, clases y etiquetas, para poder captar la información necesaria y realizar la ejecución de código pertinente.

Para profundizar en detalle el concepto de DOM y sus características, puedes consultar el documento **Material Apoyo Lectura - El estándar DOM**, ubicado en "Material Complementario".

Métodos de uso común para Manipular el DOM

El acceso a elementos del DOM puede realizarse de diferentes maneras en el código JavaScript, en función de nuestras necesidades.

Estos accesos pueden realizarse mediante:

- **Id:** El acceso mediante el atributo id es uno de los más utilizados y de fácil uso, puesto que es un identificador único de un elemento en el DOM.

```
<div id="contenedor">  
  <p id="parrafo">Hola soy un párrafo.</p>  
</div>
```

```
var parrafo = document.getElementById("parrafo");
```

En este caso, si existe un elemento con el id `parrafo`, se retornará como objeto. De lo contrario, el valor retornado será null. Este resultado se replica para todos los tipos de búsqueda.

- **Tag:** El acceso por tag se realiza especificando el tipo de elemento HTML. Por ejemplo, si deseamos buscar todos los elementos de tipo párrafo en el DOM (`<p>`), podríamos escribir el siguiente código:

```
<div id="contenedor">
  <p>Hola soy un párrafo.</p>
</div>
```

Usando **getElementsByTagName**, seleccionamos el elemento p.

```
var parrafos = document.getElementsByTagName("p");
```

- **Clase:** También podemos buscar un elemento mediante su atributo class en el DOM. Por ejemplo, si deseamos buscar un elemento que contenga la clase botón, podríamos escribir el siguiente código:

```
<div id="contenedor">
  <button type="button" class="boton">Soy un botón</button>
</div>
```

Usando **getElementsByClassName**, seleccionamos el elemento p.

```
var boton = document.getElementsByClassName("boton");
```

- **Selectores CSS:** Es posible buscar elementos mediante selectores CSS (de la misma forma que se realiza en el código CSS). Por ejemplo, si deseamos buscar todos los elementos <p> con la clase **parrafo**, podríamos escribir el siguiente código:

```
<div id="contenedor">
  <p class="parrafo">Hola soy un párrafo.</p>
  <p class="parrafo">Hola soy otro párrafo.</p>
</div>
```

Utilizando la instrucción **document.querySelectorAll()** podemos acceder a un selector de CSS, en este caso, a la clase denominada **parrafo**.

```
var parrafos = document.querySelectorAll(".parrafo");
```

Por lo tanto, el método **querySelectorAll** nos traería un arreglo con todos los elementos que se ajusten a nuestra búsqueda. Ahora, también disponemos del método **querySelector** que nos retornaría sólo el primer elemento que calce con nuestro patrón de búsqueda:

```
var parrafos = document.querySelector(".parrafo");
```

Ejercicio guiado: Acceso a elementos del DOM

Ahora desarrollaremos un ejercicio para poner en práctica los conocimientos que hemos ido adquiriendo sobre los elementos del DOM. Para esto, se solicita seleccionar los siguiente elementos mediante el uso del DOM:

1. El "id" con valor "contenedor".
2. El ul mediante la instrucción **getElementsByTagName**.
3. La clase con el valor "menu" mediante la instrucción **getElementsByClassName**.
4. La clase con el valor "item" mediante la instrucción **querySelectorAll**.

Para este desarrollo, se tiene como base el siguiente documento HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <div id="contenedor">
    <div class="menu">
      <ul>
        <li class="item">Item 1</li>
        <li class="item">Item 2</li>
        <li class="item">Item 3</li>
      </ul>
    </div>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

- **Paso 1:** Crea una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.
- **Paso 2:** En el index.html debes escribir el código proporcionado al inicio del ejercicio.
- **Paso 3:** En el archivo script.js, que ya se encuentra enlazado desde el index.html, agregaremos la instrucción necesaria para poder seleccionar el elemento indicado

en el HTML con el **id** igual a contenedor (punto 1) mediante las instrucciones “document.getElementById()” y almacenamos este resultado en una variable para futuras operaciones y procesos que veremos más adelante, seguidamente mostramos el resultado de la variable en un **console.log**.

```
var idContenedor = document.getElementById("contenedor");  
console.log(idMenu);
```

- **Paso 4:** Al ejecutar el código anterior, el resultado sería:

```
<div id="contenedor">...</div>
```

Como se puede apreciar en el resultado anterior, la instrucción document.getElementById(), trae todo el nodo del elemento HTML mediante el DOM.

- **Paso 5:** En el archivo script.js, agregaremos la instrucción necesaria para poder seleccionar el elemento indicado en el HTML con la etiqueta mediante las instrucciones “document.getElementsByTagName()” (punto 2) y almacenamos este resultado en una variable, seguidamente mostramos el resultado por consola:

```
var elementoUl = document.getElementsByTagName("ul");  
console.log(elementoUl);
```

- **Paso 6:** Al ejecutar el código anterior, el resultado sería:

```
HTMLCollection { 0: ul, length: 1 }
```

Como se puede apreciar en el resultado anterior, la instrucción document.getElementsByTagName() trae toda la colección de elementos “ul” mediante el DOM. En este ejemplo, como solo existe una etiqueta “ul” en el HTML, el resultado del arreglo sería que tiene un solo elemento “ul” ubicado en la posición 0.

- **Paso 7:** En el archivo script.js, agregaremos la instrucción necesaria para poder seleccionar el elemento indicado en el HTML con la clase “menu” mediante la instrucción “document.getElementsByClassName()” (punto 3) y almacenamos este resultado en una variable, mostrando el resultado por consola:

```
var menuClase = document.getElementsByClassName("menu");  
console.log(menuClase);
```

- **Paso 8:** Al ejecutar el código anterior, el resultado sería:

```
HTMLCollection { 0: div.menu, length: 1 }
```

Como se puede apreciar, la instrucción `document.getElementsByClassName()`, trae toda la colección de elementos que tengan la clase con el nombre "menu". En este ejemplo, como solo existe una etiqueta con la clase "menu" en el HTML, el resultado del arreglo es que tiene un solo elemento con esa clase ubicado en la posición 0.

- **Paso 9:** En el archivo `script.js`, agregaremos la instrucción necesaria para poder seleccionar los elemento indicado en el HTML con la clase "item" mediante la instrucción `document.querySelectorAll()` (punto 4) y almacenamos este resultado en una variable, seguidamente mostramos el resultado por consola:

```
var items = document.querySelectorAll(".item");  
console.log(items);
```

- **Paso 10:** Al ejecutar el código anterior, el resultado sería:

```
NodeList(3) [ li.item, li.item, li.item ]
```

Como se puede apreciar en el resultado anterior, la instrucción `document.querySelectorAll()`, trae toda la colección de elementos que tengan la clase con el nombre "item". En el ejemplo como existen tres elementos con la clase "item" en el HTML, el resultado del arreglo es que tiene tres elementos con esa clase, mostrando las etiquetas donde se encuentran.

Ejercicio propuesto (1)

En el HTML mostrado a continuación, selecciona mediante los métodos de acceso al DOM la primera clase con el nombre "item", el id con el nombre "lista" y todas las clases con el nombre "botones". Finalmente, los elementos HTML "<p>". Mostrar por consola cada uno de los elementos seleccionados y guardados en sus respectivas variables.

```
<div id="contenedor">
  <div class="item">
    <ul>
      <li id="lista"><p>Item 1</p></li>
      <li class="items"><p>Item 1</p></li>
      <li class="items"><p>Item 1</p></li>
    </ul>
  </div>
  <div class="item">
    <ul>
      <a href="#" class="botones">Enlace 1</a>
      <a href="#" class="botones">Enlace 2</a>
      <a href="#" class="botones">Enlace 3</a>
    </ul>
  </div>
</div>
```

Cambiando elementos del DOM

Para obtener el contenido de un elemento, podemos hacer uso de la propiedad `innerHTML` a través del acceso de un objeto. Por ejemplo, dado el siguiente párrafo definido en el código HTML:

```
<p id="parrafo">Hola a todos</p>
```

El acceso al contenido se realiza omitiendo la asignación a `innerHTML` y almacenando esto en una variable. Ejemplo:

```
var parrafo = document.getElementById("parrafo").innerHTML;
console.log(parrafo);
```

Con esto, cambiamos su contenido de "Hola a todos" por "Soy un párrafo". Podemos configurar su contenido de la siguiente manera en el código JavaScript:

```
document.getElementById("parrafo").innerHTML = "Soy un párrafo";
```

La manipulación de elementos que acabamos de ver, se puede aplicar muy bien para elementos, ya que su texto visible se define entre llaves, como en el caso de los párrafos, etiquetas H1, H2, entre otras. En cambio, para elementos como botones, donde el texto

visible está presente en un atributo, el procedimiento es distinto. Por ejemplo, en el siguiente botón:

```
<input type="button" value="Accion" id="btnAccion">
```

Podemos ver que el texto está presente en el atributo value, si quisiéramos manipular este texto, podríamos escribir el siguiente código JavaScript:

```
document.getElementById("btnAccion").value = "Boton";
```

Primero accedimos al objeto con `getElementById` y luego al atributo `value`, asignando el nuevo texto llamado `Boton`. Así como manipulamos el texto del botón, podemos manipular los diferentes atributos de los elementos HTML que quisiéramos. Por ejemplo, podríamos cambiar el `id` del botón mediante el siguiente código:

```
document.getElementById("btnAccion").id = "nuevoId";
```

También, cambiar el botón por una caja de texto con lo siguiente:

```
document.getElementById("btnAccion").type = "text";
```

Ahora, disponemos de un método específico para configurar atributos de los elementos del DOM: **setAttribute**, el cual recibe como parámetro el nombre del atributo que se desea configurar junto con el valor que se le asociará. En el caso del botón con el atributo **id** y valor **btnAccion**, podríamos configurar su propiedad **style** para, por ejemplo, cambiar su color de fondo:

```
document.getElementById('btnAccion').setAttribute('style',  
'background-color: blue');
```

Así como podemos modificar elementos y propiedades mediante elementos del DOM, también es posible añadir elementos y escribir un documentos HTML. Puedes profundizar sobre este contenido en el documento **Material Apoyo Lectura - Gestionando elementos del DOM**, ubicado en "Material Complementario". Ahí podrás ver en detalle elementos teóricos y prácticos de este contenido.

Ejercicio guiado: Cambiando elementos del DOM

Se requiere modificar el texto asociado al elemento `<p>` que posea un `id` con el nombre `"textoSaludo"` a `"Hola, este párrafo fue modificado"`. Igualmente, modificar los valores del elemento `<input>` con `id` igual a `"entradaUno"`, por el `value` igual a `"Clic Aqui"`, el `id` por `"clicUno"` y el tipo `type` por `"button"`. Además, agregar el atributo `style` con la propiedad `"color: red; background-color: green"`. Implementar las instrucciones necesarias para acceder al DOM y modificar los valores mencionados desde el siguiente documento HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <div id="contenedor">
    <div class="menu">
      <p id="texto">Soy un párrafo en un documento HTML</p>
      <p id="textoSaludo">Soy un párrafo en un documento HTML</p>
      <input value="Correo Electrónico" id="entradaUno" type="email"/>
      <input value="Entrada de datos" id="entradaDos" type="email"/>
    </div>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

- **Paso 1:** En el archivo `script.js`, agregaremos la instrucción necesaria para poder seleccionar los elementos y así realizar las modificaciones solicitadas. Por lo tanto, para seleccionar el elemento `<p>` con el `id` de nombre `"textoSaludo"`, debemos utilizar el `document.getElementById()`, para modificar el texto asociado dentro de las etiquetas `<p>`, se debe utilizar el `"innerHTML"`, quien permitirá modificar ese valor:

```
document.getElementById("textoSaludo").innerHTML = "Hola, este párrafo
fue modificado";
```

- **Paso 2:** Modificaremos el elemento `input` con el `id` denominado `"entradaUno"`, en este caso se deben cambiar distintos atributos de este elemento, como el caso del `value`,

id y type. Se debe utilizar la instrucción `document.getElementById()`, más las propiedades de id, value y type para cambiar el valor existente por el nuevo valor:

```
document.getElementById("entradaUno").value = "Click Aquí";  
document.getElementById("entradaUno").type = "button";  
document.getElementById("entradaUno").setAttribute('style', 'color: red;  
background-color: green');
```

Al ejecutar el código anterior en el navegador web, el resultado sería:

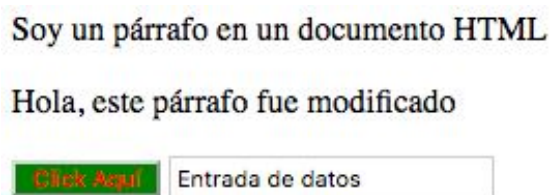


Imagen 1. Resultado HTML en el navegador después de las modificaciones.

Fuente: Desafío Latam

Ejercicio propuesto (2)

Para el HTML mostrado a continuación, seleccionar y modificar el texto asociado al id con nombre "lista1" mediante la propiedad "innerHTML" de su valor original a "Ingrese el nombre: ", igualmente seleccionar y modificar los valores del elemento con el "id=lista2", modificar el tipo "type" de su valor inicial por "submit", el "value" de su valor original a "Enviar Formulario", el id de su valor original a "envioFormulario" y finalmente agregar un estilo donde el texto cambie de su color original a azul y el color de fondo a gris (style="color: blue, background-color: gray").

```
<div id="contenedor">  
  <div class="item">  
    <ul>  
      <p id="lista1">Item 1</p>  
      <input type="text">  
      <input id="lista2" value="Enlace Archivo externo" type="text">  
    </ul>  
  </div>  
</div>
```

Eventos del DOM para reaccionar a elementos HTML

Competencias

- Codificar un script que permita la selección y manipulación de elementos del DOM aplicando listener.
- Codificar una rutina en JavaScript utilizando el método `preventDefault()` para cancelar el comportamiento por defecto de un objeto de HTML.

Introducción

Cuando visitamos un sitio web y hacemos click en algún botón o menú este reacciona y nos abre ventanas nuevas, modals o hace que alguna sección cambie de color o tamaño, estas acciones son eventos que puedes realizar con JavaScript. Por otro lado, los eventos por sí solos no son suficientes, es necesario asociar funciones para agregarle lógica a nuestros sitios y así construir páginas con una experiencia de usuario enriquecida.

A continuación, aprenderemos cómo añadir comportamientos a elementos del DOM y también cómo manipular elementos del DOM a partir de otros eventos por ejemplo clicks sobre botones, entre otros. Este es el primer paso para utilizar las herramientas que nos provee JavaScript para capturar y responder ante cualquier evento que se produzca en el sitio.

Eventos

El estándar HTML DOM incluye eventos que nos permiten reaccionar mediante JavaScript, a eventos HTML ya definidos. Esto funciona como una suerte de observador, el cual está pendiente de la ejecución de eventos HTML, como lo son el hacer click sobre un botón (click), pasar el mouse sobre una caja de texto (mouseover), cuando se cargan los elementos en el body (onload), entre otros.

Estos observadores son conocidos como listeners. Para agregar un listener a un elemento del DOM, se debe hacer uso del método `addEventListener`, que puede recibir como parámetros:

- **Evento:** Nombre del evento al cual se asociará el listener (click, mouseover, onload, etc).
- **Función:** Una función que se ejecutará cuando se dispare el evento señalado.
- **Propagación del evento (opcional):** Valor booleano (true o false) indicando el orden en que se propagará el evento, desde su elemento más interno al externo o viceversa.

La sintaxis para declarar un listener a un elemento del DOM sería la siguiente:

```
element.addEventListener(evento, funcion, propagacion);
```

Cabe señalar que no sólo es posible asociar un listener a un elemento, sino que pueden ser varios. Ahora, con el siguiente código HTML:

```
<div id="box" style="width:100px;height:100px;background-color:red">  
  Haz click sobre mí  
</div>
```

Podemos asociar el evento click sobre sí mismo, para que despliegue un alert utilizando el siguiente código Javascript:

```
let box = document.getElementById('box');  
box.addEventListener('click', function(){  
  alert('click sobre la caja');  
});
```

Si ejecutamos el código anterior y hacemos click sobre el div de color rojo que generamos, podremos ver el mensaje **click sobre la caja**, como se ve en la siguiente imagen.



Imagen 2. Resultado al hacer click en el div rojo

Fuente: Desafío Latam

El ejemplo anterior se puede refactorizar de la siguiente manera:

```
function miFuncionAlerta(){  
    alert('click sobre la caja');  
}  
  
let box = document.getElementById('box');  
box.addEventListener('click', miFuncionAlerta);
```

Otro ejemplo del uso de este método sería controlar que el puntero del mouse se posicione por encima del div creado y luego cuando desaparezca de su perímetro, como se hace con el siguiente código:

```
box.addEventListener('mouseover', function(){  
    box.innerHTML = 'El puntero está por encima';  
});  
  
box.addEventListener('mouseout', function(){  
    box.innerHTML = 'El puntero ya no está por encima';  
});
```

Ahora, al tratar de realizar dichas acciones, podrás ver como el contenido del div con el id box cambiará su mensaje en función de si pasamos el mouse por encima o lo sacamos. Además, como indicamos anteriormente los listener son acumulativos y no desaparecen por agregar uno u otro control de eventos sobre un elemento del DOM.

Otro caso muy común en el desarrollo de sitios web es el trabajo con formularios, veamos un par de ejemplos y como JavaScript nos ayuda a manipular su información.

Conocer y aplicar listener sobre elementos del DOM

En el punto anterior, se detallaron los eventos que ocurren en un documento y nuestro navegador web, cómo podemos captarlos, trabajar con ellos y ejecutar acciones en específico cuando estos eventos son activados. Por lo tanto, JavaScript permite agregar listener (escuchadores) de eventos a los elementos que forman parte del DOM, una de estas instrucciones que permiten captar y actuar en consecuencia a un evento es la instrucción **addEventListener**, la cual tiene dos elementos claves, el **evento** (se debe indicar cual es el evento que se quiere escuchar) y la **función** que se ejecutará cuando se active el evento.

Ejercicio guiado: Rescatar el valor del input

Se posee un sitio web donde existe un buscador, con ayuda de listener vamos a rescatar el valor que escriba el usuario en un campo creado con un elemento input. Además, se contará con un botón donde el usuario podrá hacer clic y mostrar el texto escrito en el input dentro de una etiqueta <p> con la clase denominada "resultado". Cuentas con el documento HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <div class="contenedor">
    <div class="buscador">
      <input class="input-a-buscar" type="text" name="buscador">
      <button id="boton" type="button">Buscar</button>
    </div>
    <p class="resultado"></p>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

- **Paso 1:** Revisar parte por parte el código de JavaScript que nos permite recoger el valor ingresado en el formulario. Lo primero que se necesita es acceder al elemento **button** para luego hacer uso del listener **addEventListener**. Entonces se debe guardar

el elemento que trae la instrucción `getElementById` en una variable, para poder aplicar el `addEventListener` directamente a esa variable con el evento del tipo “click”, quedando el código:

```
var miBtn = document.getElementById("boton");
miBtn.addEventListener('click',function(){
  //cuerpo de la función anónima que se ejecuta al hacer click
});
```

- **Paso 2:** Ahora una vez que el usuario haga click en el botón, debemos tomar el valor ingresado en el elemento input y agregarlo a nuestro párrafo con clase resultado al final de la estructura HTML, para esto hacemos uso de los selectores `querySelector`, el cual, permite buscar mediante los selectores de CSS el elemento donde debemos agregar y mostrar el texto ingresado por el usuario:

```
var texto = document.querySelector(".input-a-buscar");
document.querySelector(".resultado").innerHTML = "Estas buscando: " +
texto.value;
```

- **Paso 3:** Ahora si unimos todo el código, quedaría el script de esta manera:

```
var miBtn = document.getElementById("boton");
miBtn.addEventListener('click',function(){
  var texto = document.querySelector(".input-a-buscar");
  document.querySelector(".resultado").innerHTML = "Estas buscando: " +
texto.value;
});
```

- **Paso 4:** Al ejecutar el código anterior en nuestro navegador, el resultado obtenido al ingresar perros en el campo, sería:

Estas buscando: perros

Imagen 3. Buscador web
Fuente: Desafío Latam

Con el ejercicio realizado anteriormente, se logró conocer y aplicar “listener” sobre elementos en específico del DOM sobre la misma función que lleva la instrucción del `addEventListener`.

Ejercicio propuesto (3)

Desarrollar un programa en JavaScript que permita leer el número ingresado por el usuario en un campo creado, con un elemento input del tipo "number" y un id con el nombre "entrada", solo cuando se active el enlace mediante un click del usuario. El lugar para mostrar el mensaje se encuentra al final de la estructura HTML y posee la clase con nombre "resultado", dicho mensaje debe indicar, por ejemplo: "El número ingresado fue el 23". Para ello cuentas con el siguiente HTML:

```
<div>
  <div>
    <p>Ingresa un número</p>
    <input type="number" name="numero" id="entrada">
    <a href="#" type="button" id="evento">Click Aquí.</a>
  </div>
  <div><p class="resultado"></p></div>
</div>
```

Ejecutar funciones creadas de JavaScript al enviar un formulario

Hasta el momento, se trabajó con el evento "click" y la instrucción `addEventListener` para captar el evento y ejecutar funciones sobre la misma instrucción o fuera de ella. Pero así como existe el evento "click" existen otros tipos de eventos, como el del tipo "submit", el cual, viene asociado por defecto al elemento "form" de HTML. Es decir, cada vez que el usuario haga click en un botón del tipo "submit" dentro de un formulario, este evento se ejecutará por sí solo. Por lo tanto, vamos a aprovechar este evento, capturarlo y ejecutar una función para procesar los datos de un formulario.

Para profundizar en el uso de funciones utilizando el evento click, así como ejercitar casos para escribir un código más ordenado, limpio y fácil de entender, puedes consultar el documento **Material Apoyo Lectura - Ejecutar funciones creadas de JavaScript al hacer click en un botón**, ubicado en "Material Complementario". Ahí podrás ver en detalle elementos teóricos y prácticos de estos contenidos.

Ejercicio guiado: Login

Desarrollar un ejercicio con todo lo estudiado hasta el momento con respecto a los eventos en el DOM. En el sitio web se encuentra un formulario que solicita a los usuarios ingresar el correo electrónico y contraseña. Con ayuda de listener vamos a rescatar los valores que escriba el usuario dentro del formulario. Además, se contará con un botón donde el usuario podrá hacer clic para procesar los datos del login y mostrar el texto escrito en el input dentro de una etiqueta <p> con la clase denominada resultado. Indicar en el mensaje, por ejemplo: "Bienvenido usuario@usuario.com". Documento HTML donde se encuentra el formulario:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <div>
    <form id="formulario" name="formulario">
      <div>
        <label for="email">Email</label>
        <input class="email" type="email" >
      </div>
      <div>
        <label for="password">Contraseña</label>
        <input class="password" type="password" >
      </div>
      <button type="submit" >Ingresar</button>
    </form>
  </div>
  <div>
    <p class="resultado"></p>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

- **Paso 1:** Acceder al elemento formulario primeramente y luego le agregamos un listener, en este caso, usaremos el evento "submit". Los submit son eventos propios

de los formularios y se utilizan para enviar la información que es introducida en estos, igualmente que en el ejemplo anterior, activaremos una función externa que llamaremos "login", quedando el código:

```
let form = document.getElementById( "formulario" );  
form.addEventListener( "submit", login);
```

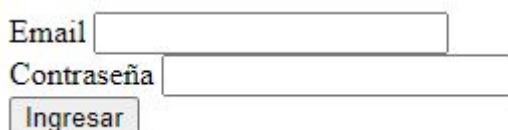
- **Paso 2:** Ahora en la función login, debemos leer los datos ingresados por el usuario y dar la bienvenida como usuario registrado:

```
function login(){  
  var email = document.querySelector(".email");  
  var password = document.querySelector(".password");  
  document.querySelector(".resultado").innerHTML = `Bienvenido  
${email.value}`;  
};
```

- **Paso 3:** Unir todos los trozos de código en uno solo:

```
function login(){  
  var email = document.querySelector(".email");  
  var password = document.querySelector(".password");  
  document.querySelector(".resultado").innerHTML = `Bienvenido  
${email.value}`;  
};  
  
let form = document.getElementById( "formulario" );  
form.addEventListener( "submit", login);
```

- **Paso 4:** Al ejecutar el código anterior, agregando un correo, una contraseña y haciendo un click en el botón de enviar, el cambio ocurre muy rápidamente, debido a que la página se recarga nuevamente ella sola, limpiando los campos con los valores ingresados y desapareciendo el mensaje mostrado en la parte inferior del documento. Esto se debe al comportamiento por defecto de los formularios con el evento submit, que recargan la página donde se encuentran automáticamente.



Formulario de ingreso visualizado:

Email

Contraseña

Imagen 4. Formulario de ingreso
Fuente: Desafío Latam

Ejercicio propuesto (4)

Desarrollar un programa en JavaScript que permita leer el nombre, correo electrónico y clave ingresados por el usuario en campos separados creados con elementos input. Cuando se active el evento del formulario mediante un click del usuario en un botón destinado para tal fin, debe aparecer un mensaje en una ventana emergente e indicar, por ejemplo: *"Hola Juan, tu correo electrónico es usuario@usuario.com y tu clave es 123456, bienvenido!!!"*. Puedes utilizar como base el código HTML del ejercicio anterior.

Ejercicio guiado: Cancelar el comportamiento por defecto de un objeto

Como vimos en el Ejercicio guiado: Login, existen cambios que se ejecutan por defecto y que hacen que la página tenga un comportamiento distinto al que esperamos. Por ende, veamos cómo cancelar este comportamiento:

- **Paso 1:** Partiendo del ejercicio anterior en el **paso 3**, realizamos unas pequeñas modificaciones para poder apreciar mejor el evento "submit" del formulario, agregando un alert después de recibir los datos para mostrar el correo electrónico:

```
function login(){  
  var email = document.querySelector(".email");  
  var password = document.querySelector(".password");  
  alert("Ingreso exitoso: "+email.value);  
};  
  
let form = document.getElementById( "formulario" );  
form.addEventListener( "submit", login);
```

- **Paso 2:** Ejecuta nuevamente el código anterior en el navegador web ingresando como correo electrónico "usuario@usuario.com" y el resultado en pantalla sería:



Imagen 5. Resultado en el navegador web.
Fuente: Desafío Latam

Si te diste cuenta después que salta el alert con el mensaje, la página se vuelve a cargar borrando los datos por completo del formulario; esto no es un error si no una condición no controlada cuando un formulario hace submit, ya que por defecto está preparado para ejecutar una acción (generalmente un llamado a una ruta del backend). Para evitar comportamientos por defecto, haremos uso de la función **preventDefault()**, la cual permite cancelar la acción o respuesta por defecto que tiene un elemento.

- **Paso 3:** Aplicamos ahora el **preventDefault()** al paso anterior dentro de la función "login", agregando el parámetro "event" a la función para que pueda ser leído y utilizado en conjunto con el preventDefault, quedando.

```
function login(event){  
  event.preventDefault();  
  var email = document.querySelector(".email");  
  var password = document.querySelector(".password");  
  alert("Ingreso exitoso: "+email);  
};  
  
let form = document.getElementById( "formulario" );  
form.addEventListener( "submit", login);
```

- **Paso 4:** Ahora si pruebas, notarás que ya no se redirecciona y el formulario sigue manteniendo los valores ingresados antes de hacer un click en el botón. Por lo tanto, el evento submit en el formulario no realiza su comportamiento por defecto. Si te das cuenta, ahora en vez de tener el código dentro del listener, llamamos a una función que ejecuta el login.



Imagen 6. Resultado en el navegador web.
Fuente: Desafío Latam

Ejercicio propuesto (5)

Al ejercicio propuesto número 4, agrégale la función **preventDefault()** para evitar el comportamiento por defecto del formulario.

Ejercicio guiado: Validar un formulario utilizando JavaScript

Un uso común que vemos día a día en la web son las validaciones que se aplican a los inputs de un formulario. Por ejemplo, cuando te quieres registrar en un sitio web y la contraseña debe cumplir una serie de reglas o cuando en un buscador no ingresas nada y aprietas el botón “buscar”, aparece una ventana con un error por no ingresar datos.

Para ejemplificar este caso, realizaremos un ejercicio partiendo del código HTML del *Ejercicio guiado: Rescatar el valor del input*.

- **Paso 1:** En el index.html, se agrega la estructura del documento mencionado.
- **Paso 2:** Revisar el código de JavaScript que nos permite recoger el valor ingresado en el formulario. Lo primero que se necesita es acceder al elemento **button** para luego hacer uso del listener **addEventListener**. Entonces se debe guardar el elemento que trae la instrucción `getElementById` en una variable, para poder aplicar el `addEventListener` directamente a esa variable con el evento del tipo “click”:

```
var miBtn = document.getElementById("boton");  
miBtn.addEventListener('click',function({}));
```

- **Paso 3:** Ahora una vez que el usuario haga click en el botón, debemos tomar el valor ingresado en el elemento input y agregarlo a nuestro párrafo con clase resultado al final de la estructura HTML, para esto hacemos uso del selector `querySelector`:

```
var texto = document.querySelector(".input-a-buscar");  
document.querySelector(".resultado").innerHTML = "Estas buscando: " +  
texto.value;
```

- **Paso 4:** Ahora si unimos todo el código, quedaría el script de esta manera:

```
var miBtn = document.getElementById("boton");  
  
miBtn.addEventListener('click',function(){  
  var texto = document.querySelector(".input-a-buscar");  
  document.querySelector(".resultado").innerHTML = "Estas buscando: " +  
  texto.value;  
});
```

- **Paso 5:** Este código cumple con el objetivo, pero si uno presiona el botón sin antes introducir el texto este sigue funcionando y muestra un resultado vacío. Para prevenir esta situación, validamos que para mostrar el mensaje debe haber ingresado cualquier texto en el input:

```
var miBtn = document.getElementById("boton");

miBtn.addEventListener('click',function(){
  var texto = document.querySelector(".input-a-buscar");

  if(texto.value !== "") {
    document.querySelector(".resultado").innerHTML = "Estas buscando: "
+ texto.value;
  };
});
```

- **Paso 6:** La validación está correcta, pero ahora se siente como que el botón no hace nada, siempre que estamos validando algo es importante indicarle al usuario lo que está pasando. Para mostrar un mensaje de error, modificaremos nuestro HTML agregando un `<p class="error"></p>`:

```
<div class="contenedor">
  <div class="buscador">
    <input class="input-a-buscar" type="text" name="buscador">
    <button id="boton" type="button">Buscar</button>
  </div>
  <p class="error" style="color: red"></p>
  <p class="resultado"></p>
</div>
```

- **Paso 7:** Ahora debemos agregar la lógica para que cuando el usuario no ingrese un texto se le indique y pueda así solventar el error agregando el texto antes de buscar:

```
if (texto.value !== "") {
  document.querySelector(".resultado").innerHTML = "Estas buscando: " +
texto.value;
} else {
  document.querySelector(".error").innerHTML = "Para poder buscar debes
ingresar una palabra";
};
```

- **Paso 8:** Si ejecutas todo el código anterior, podemos observar en pantalla lo siguiente cuando hagas un click en el botón buscar sin ingresar ningún tipo de texto:



Un formulario de búsqueda simple con un campo de entrada de texto rectangular y un botón rectangular a su derecha con el texto "Buscar".

Para poder buscar debes ingresar una palabra

Imagen 7. Mensaje de error
Fuente: Desafío Latam

- **Paso 9:** Como te habrás dado cuenta, el código ahora muestra un mensaje cuando no se ingresa una palabra, pero si el usuario ahora escribe cualquier texto en el campo de búsqueda y hace un click sobre el botón buscar, el mensaje de error se mantiene. Para corregir este detalle, se puede enviar una cadena de texto vacía al innerHTML del elemento cuando la condición se cumpla, de lo contrario se envía el mensaje de error en sí, alternando el mensaje dependiendo del resultado de la condición de validación. Ahora nuestro buscador ha quedado validado:

```
if(texto.value !== "") {  
    document.querySelector(".resultado").innerHTML = "Estas buscando: " +  
    texto.value;  
    document.querySelector(".error").innerHTML = "";  
}else {  
    document.querySelector(".resultado").innerHTML = "";  
    document.querySelector(".error").innerHTML = "Para poder buscar debes  
    ingresar una palabra";  
}
```

Ejercicio propuesto (6)

Un sitio web necesita implementar un pequeño formulario para registrar el correo de los usuarios que deseen recibir notificaciones. Para ello, debes validar el ingreso del dato en el campo del correo antes de poder procesar la información, es decir, el campo no puede estar sin datos, por esta razón, cuando el usuario haga un click sobre el botón el campo debe revisarse y verificar que contenga dato.

Esta condición se debe mostrar en un mensaje en el caso de no existir dato alguno en el campo indicando que "debe existir un correo electrónico para poder registrarse", de existir dato se debe mostrar, ejemplo: "tu correo electrónico usuario@correo.com fue registrado con éxito." todo esto mediante un alert.

```
<div class="contenedor">
  <form id="formulario" name="formulario">
    <div>
      <label for="correo">Correo Electrónico: </label>
      <input class="correo" type="email" >
    </div>
    <button type="submit">Registrar</button>
  </form>
</div>
```


Expresiones Regulares

Competencias

- Reconocer las principales características de las expresiones regulares para identificar sus ventajas y contextos de uso.
- Aplicar expresiones regulares básicas sobre cadenas de texto para validar la estructura de la información recibida.

Introducción

Una de las aristas a tomar en cuenta a la hora de desarrollar aplicaciones y más aún cuando necesitamos manipular datos desde un lado a otro, es poder realizar validaciones de los mismos. Para nuestra suerte, contamos con mecanismos muy eficientes para poder realizar este tipo de tareas.

Estos mecanismos nos facilitarán realizar labores de validación en datos ingresados por parte del usuario. Por ejemplo, cuando ingresamos datos en un formulario, como el rut o correo electrónico, existen lógicas que permiten determinar los patrones válidos posibles y optimizar la cantidad de líneas de código de nuestros proyectos, haciéndolo más eficiente.

¿Qué son las Expresiones Regulares (RegExp)?

Las expresiones regulares (del inglés Regular Expressions o conocidas como RegExp) son objetos que describen patrones de caracteres y se utilizan como patrones de búsqueda. Son ampliamente utilizados a la hora de realizar validaciones de cadenas de texto, ya que suelen tener una sintaxis muy corta y su utilización es muy sencilla a la hora de validar.

Sintaxis

La sintaxis de una expresión regular se comprende por el patrón definido, junto con sus modificadores:

```
/patrón/modificadores;
```

Un ejemplo de algunas expresiones regulares comúnmente usadas en JavaScript son:

```
/[a-zA-Z]/gim  
/[A-Z0-9._%+-]+@[A-Z0-9-]+\.[A-Z]/gim
```

Comprendiendo y validando datos con Expresiones Regulares

Para validar datos con expresiones regulares, podemos hacerlo mediante el uso del método `match` (aunque también existen otros métodos) presente en cada objeto que contenga una cadena de texto como valor, es decir, un `string`. Este método retorna un arreglo (dependiendo del modificador utilizado) en el caso de encontrar alguna coincidencia con la coincidencia hallada. De lo contrario, se retornará **null**.

Por ejemplo, si quisiéramos validar que una cadena de texto que contenga la palabra `gato`, podríamos generar el siguiente patrón:

```
/gato/i
```

Este patrón se compone de lo siguiente:

- **/gato/**: Es el patrón que se utilizará para buscar una coincidencia.
- **i**: Corresponde al modificador que se aplicará al patrón. En este caso, este modificador establece que la búsqueda del patrón no será afectada por mayúsculas o minúsculas (case insensitive).

Para este caso y de cumplirse una coincidencia con esta expresión regular, obtendremos en JavaScript el siguiente arreglo como resultado:

```
["gato"]
```

Para profundizar el concepto de modificadores, conocer cuáles existen y cómo aplicarlos, puedes consultar el documento **Material Apoyo Lectura - Modificadores de Expresiones Regulares**, ubicado en “Material Complementario”.

Ejercicio guiado: Validar ingreso de datos

Validar el ingreso de datos en un buscador de animales, por lo que solo debe aceptar las palabras “perro” y “gato”. De lo contrario, no puede permitir la búsqueda, generando un mensaje de error en un alert. Para este ejercicio se cuenta con el documento HTML que tendrá disponible el usuario.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <label for="animal">Buscador de razas de Perros y gatos</label>
  <input type="text" class="animal" placeholder="Ingresa el texto
aquí">
  <input type="button" id="buscar" value="Buscar">
  <script src="script.js"></script>
</body>
</html>
```

- **Paso 1:** En el archivo script.js, será donde se agrega toda la lógica de programación solicitada en el enunciado, por lo tanto, lo primero que se necesita es acceder al botón de búsqueda agregando un escucha, luego, agregar una función externa para que poder capturar los datos ingresados por el usuario y aplicar el proceso de validación mediante las expresiones regulares, quedando el código:

```
var buscar = document.getElementById("buscar");  
buscar.addEventListener('click',validar);
```

- **Paso 2:** Ahora una vez que el usuario haga click en el botón, debemos tomar los valores ingresados en el elemento input, para esto hacemos uso de la instrucción `querySelector`, igualmente creamos las dos expresiones regulares que utilizaremos dentro de una estructura condicional "if" en conjunto con el método `match()`, que busca un string para una coincidencia con una expresión regular definida. Es importante mencionar que las expresiones regulares se le agrega el modificador "i" que permite omitir entre mayúsculas y minúsculas:

```
function validar(){  
  var animal = document.querySelector(".animal").value;  
  var patron1 = /gato/i;  
  var patron2 = /perro/i;  
  if (animal.match(patron1) || animal.match(patron2)){  
    alert("Palabra ingresada permitida");  
  } else {  
    alert("La palabra ingresada no es permitida");  
  }  
};
```

- **Paso 3:** Ahora si unimos todo el código JavaScript, quedaría el script de esta manera:

```
var buscar = document.getElementById("buscar");  
buscar.addEventListener('click',validar);  
  
function validar(){  
  var animal = document.querySelector(".animal").value;  
  var patron1 = /gato/i;  
  var patron2 = /perro/i;  
  if (animal.match(patron1) || animal.match(patron2)){  
    alert("Palabra ingresada permitida");  
  } else {  
    alert("La palabra ingresada no es permitida");  
  }  
};
```

- **Paso 4:** Al ejecutar el código anterior en nuestro navegador, el resultado obtenido al ingresar perro o gato en el campo requerido, sería:

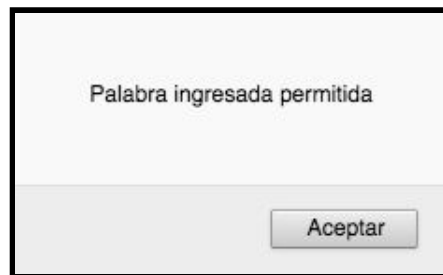


Imagen 8. Resultado final del ejercicio en el navegador web.
Fuente: Desafío Latam

- **Paso 5:** Al ingresar una palabra no permitida por el validar, el mensaje sería:

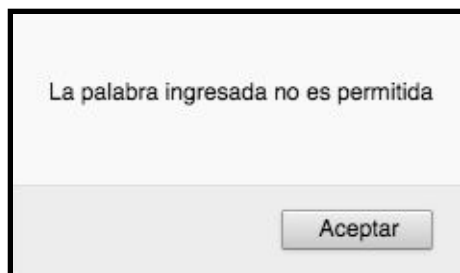


Imagen 9. Resultado final del ejercicio en el navegador web.
Fuente: Desafío Latam

Ejercicio propuesto (7)

Para un formulario donde se solicita al usuario ingresar una palabra, la cual puede ser: "JavaScript, NodeJS o VueJS". Crear una función que se active mediante un evento que permita validar que el texto ingresado por el usuario fue uno de los tres mencionados anteriormente. El mensaje debe ir en un alert e indicar: "Palabra permitida" o "Palabra no permitida", dependiendo del caso. El formulario es el siguiente:

```
<label for="palabra">Ingresa la palabra a buscar</label>  
<input type="text" class="palabra" placeholder="Ingresa el texto aquí">  
<input type="button" id="buscar" value="Buscar">
```

Corchetes en Expresiones Regulares

Los corchetes en expresiones regulares nos ayudan a encontrar un rango de caracteres que definamos dentro de ellos:

- **[abc]**: Sólo se obtienen los caracteres definidos entre corchetes.

```
const patron = /[abc]/gim;  
const texto = 'a b c ';  
console.log(texto.match(patron));
```

El resultado del código anterior es:

```
(3) ["a", "b", "c"]
```

- **[^xyz]**: Encuentra cualquier caracter que no coincida con los caracteres al interior de los corchetes:

```
const patron = /^[^xyz]/gim;  
const texto = 'a y c x ';  
console.log(texto.match(patron));
```

El resultado del código anterior sería:

```
(2) ["a", "c"]
```

Nota: dentro de los corchetes se pueden definir más letras e inclusive números para hallar patrones de búsqueda de acuerdo a nuestras necesidades.

Por ejemplo, si quisiéramos encontrar todas las coincidencias de los caracteres **s t y 1 2 3**, podríamos disponer del siguiente código:

```
const patron = /[sti123]/gim;  
const texto = 'subaru wrx sti 3 2 1 0';  
console.log(texto.match(patron));
```

Con ésto, conseguimos el siguiente resultado:

```
(7) ["s", "s", "t", "i", "3", "2", "1"]
```

- **[a-zA-Z]:** Encuentra cualquier carácter entre los rangos de la letra a hasta la z y desde A hasta Z. Es decir, válida cualquier carácter que sea una letra. Por ejemplo, con el siguiente código:

```
const patron = /[a-zA-Z]/gim;  
const texto = 'todo a excepcion del numero 1';  
console.log(texto.match(patron));
```

Obtendremos el siguiente resultado:

```
(23) [ "t", "o", "d", "o", "a", "e", "x", "c", "e", "p", "c", "i", "o",  
"n", "d", "e", "l", "n", "u", "m", "e", "r", "o" ]
```

También sería posible validar sólo rangos numéricos:

```
const patron = /[0-6]/gim;  
const texto = 'mi numero es el 761';  
console.log(texto.match(patron));
```

En este caso, nuestro resultado sería el siguiente:

```
(2) ["6", "1"]
```

Para conocer en profundidad otros aspectos de las expresiones regulares, puedes consultar el documento **Material Apoyo Lectura - Metacaracteres en Expresiones Regulares**, ubicado en “Material Complementario”, donde podrás ver en detalle cómo los metacaracteres nos aportan una forma abreviada de interpretar un patrón de búsqueda.

Ejercicio guiado: Corchetes en la expresión regular

Validar el ingreso de datos en un campo de texto que permita el ingreso de caracteres alfanuméricos combinados, más no solo numéricos. De lo contrario, no puede permitir el envío de la información, generando un mensaje de error en un alert. Para este ejercicio se cuenta con el documento HTML que tendrá disponible el usuario.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <label for="lugar">Ingresa tu dirección: </label>
  <input type="text" class="lugar" placeholder="Ingresa la dirección">
  <input type="button" id="enviar" value="Enviar">
  <script src="script.js"></script>
</body>
</html>
```

- **Paso 1:** En el archivo script.js, será donde se agrega toda la lógica de programación solicitada en el enunciado, por lo tanto, lo primero que se necesita es acceder al botón de búsqueda agregando un escucha, luego, agregar una función externa para poder capturar los datos ingresados por el usuario y aplicar el proceso de validación mediante las expresiones regulares, quedando el código:

```
var enviar = document.getElementById("enviar");
enviar.addEventListener('click', validar);
```

- **Paso 2:** Ahora una vez que el usuario haga click en el botón, debemos tomar los valores ingresados en el elemento input, para esto hacemos uso de la instrucción querySelector, igualmente creamos las expresiones regulares que utilizaremos dentro de una estructura condicional "if" en conjunto con el método match (). En este caso la expresión regular permitirá texto alfanumérico entre mayúsculas y minúsculas, pero un texto con solo caracteres numéricos no lo permitirá.

```
function validar(){
  var lugar = document.querySelector(".lugar").value;
  var permitido = /[a-zA-Z]/gim;
  if (lugar.match(permitido)){
    alert("El texto ingresado es permitido");
  } else {
    alert("Solo debe ingresar caracteres alfabéticos");
  };
};
```


- **Paso 3:** Ahora si unimos todo el código JavaScript, quedaría el script de esta manera:

```
var enviar = document.getElementById("enviar");
enviar.addEventListener('click', validar);

function validar(){
    var lugar = document.querySelector(".lugar").value;
    var permitido = /[a-zA-Z]/gim;
    if (lugar.match(permitido)){
        alert("El texto ingresado es permitido");
    } else {
        alert("El texto ingresado no es permitido");
    }
};
};
```

- **Paso 4:** Al ejecutar el código anterior en nuestro navegador, el resultado obtenido al ingresar una dirección cualquiera en el campo requerido, sería:

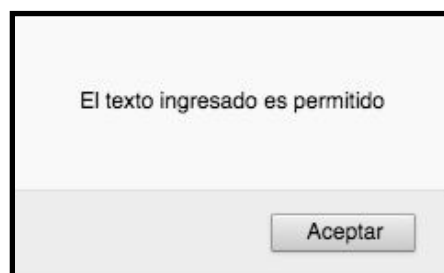


Imagen 10. Resultado final del ejercicio en el navegador web.
Fuente: Desafío Latam

Ejercicio propuesto (8)

Para un formulario donde se solicita al usuario ingresar su edad, validar el ingreso de ese campo donde se permitirán números del 0 al 9. De lo contrario, no puede permitir el envío de la información, generando un mensaje de error en un alert. Para este ejercicio se cuenta con el extracto del formulario que tendrá disponible el usuario.

```
<label for="edad">Ingresa tu edad: </label>
<input type="text" class="edad" placeholder="Ingresa la edad">
<input type="button" id="enviar" value="Enviar">
```

Utilizar regex para validar un campo de un formulario

Las expresiones regulares son muy útiles a la hora de validar formularios, por ejemplo, validar si un campo es sólo números o sólo letras, formatos especiales como correos, entre otros. En este caso, implementaremos el objeto de JavaScript RegExp, el cual, es un objeto de expresión regular con propiedades y métodos predefinidos. Entre ellos se encuentra el método "test()", quien compara una expresión dada con un patrón establecido y devuelve verdadero o falso, según el resultado.

Ejercicio guiado: Validando campos de un formulario

Validar los campos nombre, teléfono y correo de un formulario, desarrollando paso a paso cada una de las instrucciones. En este caso, el ejercicio facilita el siguiente documento HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <form action="#" name="form" id="form">
    <div>
      <label for="">Nombre</label>
      <input type="text" class="textNombre">
      <span class="errorNombre"></span>
    </div>
    <div>
      <label for="">Teléfono</label>
      <input type="text" class="textTelefono">
      <span class="errorTelefono"></span>
    </div>
    <div>
      <label for="">Email</label>
      <input type="text" class="textEmail">
      <span class="errorEmail"></span>
    </div>
    <button type="submit">Enviar</button>
    <p class="resultado"></p>
  </form>
</body>
</html>
```

```
</form>
<script src="script.js"></script>
</body>
</html>
```

- **Paso 1:** En el script.js, agregaremos un escucha al formulario con el evento del tipo submit, por lo que es necesario guardar el id del formulario en una variable y capturemos los datos dentro de la función del escucha, implementado los querySelector. Igualmente, la lógica de validación será trabajada en una función externa denominada validar(textNombre,textTelefono,textEmail) y se pasan como argumentos dentro del llamado de la función los valores capturados de los campos, en caso donde las validaciones sean correctas, retornaremos un true, igualmente, la limpieza de los campos de los errores será mediante una función externa. No te olvides de agregar la prevención de evento del formulario para que no recargues la página nuevamente.

```
let form = document.getElementById( "form" );

form.addEventListener( "submit", function ( event ) {
  event.preventDefault();
  limpiarErrores();
  let textNombre = document.querySelector(".textNombre").value;
  let textTelefono = document.querySelector(".textTelefono").value;
  let textEmail = document.querySelector(".textEmail").value;

  let resultado = validar(textNombre,textTelefono,textEmail);

  if(resultado == true) {
    exito();
  };
});
```

- **Paso 2:** Realizamos las funciones externas de limpieza, validación y de éxito. Estas funciones tendrán sus procesos por separados y serán llamados desde el escucha. Por ejemplo, la función para limpiar los errores se basará en leer el HTML de los elementos y pasarle una cadena de texto vacía, mientras que la función de éxito enviará un mensaje de validación exitosa. En el siguiente paso agregaremos el proceso de validación dentro de la función correspondiente.

```
function limpiarErrores() {
document.querySelector(".resultado").innerHTML = "";
document.querySelector(".errorNombre").innerHTML = "";
document.querySelector(".errorTelefono").innerHTML = "";
document.querySelector(".errorEmail").innerHTML = "";
};

function exito() {
    document.querySelector(".resultado").innerHTML = "Formulario pasó la validación";
};

function validar(nombre,telefono,email) {

};
```

- **Paso 3:** Comencemos con las validaciones usando expresiones regulares, para eso trabajaremos dentro de la función validar(), recibiendo como parámetros los datos ingresados en el formulario. Primero agregaremos la validación del nombre, para ello usamos el patrón `/[a-zA-Z]/gim` en el que sólo permitimos letras, entonces validamos con el método `test()` si el texto cumple o no la condición, si no mediante javascript agregamos un mensaje para que el usuario corrija. Cambiamos nuestra variable `pasamosLaValidacion` a `false` para que no se muestre el mensaje de éxito,

```
function validar(nombre,telefono,email) {

    let pasamosLaValidacion = true;
    let validacionNombre = /[a-zA-Z]/gim;

    if (validacionNombre.test(nombre) == false) {
        document.querySelector(".errorNombre").innerHTML = "Ingrese un nombre válido."
        pasamosLaValidacion = false;
    }
};
```

- **Paso 4:** Ahora aplicaremos la lógica para el teléfono y correo modificando las expresiones regulares para cada elemento. Es importante destacar que el método `test()` recibe entre los paréntesis del método el valor que queremos validar y antes del test, el patrón contra el cual validamos la expresión.

```
function validar(nombre,telefono,email) {  
  
    let pasamosLaValidacion = true;  
    let validacionNombre = /[a-zA-Z]/gim;  
  
    if (validacionNombre.test(nombre) == false) {  
        document.querySelector(".errorNombre").innerHTML = "Ingrese un  
nombre válido."  
        pasamosLaValidacion = false;  
    };  
  
    let validacionTelefono = /\d/gim;  
  
    if (validacionTelefono.test(telefono) == false) {  
        document.querySelector(".errorTelefono").innerHTML = "Ingrese un  
teléfono válido(sólo números)."  
        pasamosLaValidacion = false;  
    };  
  
    let validaciónEmail = /[A-Z0-9._%+-]+@[A-Z0-9-]+\.[A-Z]/gim;  
  
    if (validaciónEmail.test(email) == false ) {  
        document.querySelector(".errorEmail").innerHTML = "Ingrese un  
correo válido."  
        pasamosLaValidacion = false;  
    };  
  
    return pasamosLaValidacion;  
};
```

- **Paso 5:** Al unir todo el código y ejecutar el index.html en el navegador web, el resultado final debería ser el siguiente para el caso que el usuario no ingrese dato alguno:

Nombre Ingrese un nombre válido.
Teléfono Ingrese un teléfono válido(sólo
números).
Email Ingrese un correo válido.

Imagen 11. Validación de formulario
Fuente: Desafío Latam

Ejercicio propuesto (9)

Partiendo del ejercicio guiado anterior (validando campos de un formulario), agrega ahora dos nuevos campos para ser validados. En este caso, debes agregar un campo para que el usuario pueda ingresar su apellido y otro campo para ingresar la edad, ambos del tipo texto.

Resumen

En esta lectura adquirimos herramientas que nos permiten interactuar y manipular elementos de los sitios que desarrollemos, que es una de las grandes características de JavaScript.

De esta manera, ahora manejamos el concepto de DOM, los métodos más comunes para manipular y gestionar elementos. Además, podemos recibir eventos del lado del cliente, para realizar acciones con dicha información. Finalmente, abordamos las expresiones regulares y la ventaja de utilizar estos patrones, principalmente para realizar validaciones.

Ahora posees las herramientas para agregar lógica a tus sitios e interactuar con los eventos que se presenten, esto es clave en tu carrera como desarrollador/a. La invitación es a seguir practicando, resolver ejercicios y poner en práctica tus conocimientos.

Solución de los ejercicios propuestos

1. En el HTML mostrado a continuación, selecciona mediante los métodos de acceso al DOM la primera clase con el nombre "item", el id con el nombre "lista" y todas las clases con el nombre "botones". Finalmente, los elementos HTML "<p>". Mostrar por consola cada uno de los elementos seleccionados y guardados en sus respectivas variables.

```
var item = document.querySelector(".item");
var lista = document.getElementById("lista");
var botones = document.querySelectorAll('.botones');
var parrafo = document.getElementsByTagName('p');

console.log(item);
console.log(lista);
console.log(botones);
console.log(parrafo);
```

2. Para el HTML mostrado a continuación, seleccionar y modificar el texto asociado al id con nombre "lista1" mediante la propiedad "innerHTML" de su valor original a "Ingrese el nombre: ", igualmente seleccionar y modificar los valores del elemento con el "id=lista2", modificar el tipo "type" de su valor inicial por "submit", el "value" de su valor original a "Enviar Formulario", el id de su valor original a "envioFormulario" y finalmente agregar un estilo donde el texto cambie de su color original a azul y el color de fondo a gris (style="color: blue, background-color: gray").

```
document.getElementById("lista1").innerHTML = "Ingrese el nombre:";
document.getElementById("lista2").value = "Enviar Formulario";
document.getElementById("lista2").type = "submit";
document.getElementById("lista2").setAttribute('style', 'color: blue;
background-color: gray');
```

3. Desarrollar un programa en JavaScript que permita leer el número ingresado por el usuario en un campo creado, con un elemento input del tipo "number" y un id con el nombre "entrada", solo cuando se active el enlace mediante un click del usuario. El lugar para mostrar el mensaje se encuentra al final de la estructura HTML y posee la clase con nombre "resultado", dicho mensaje debe indicar, por ejemplo: "El número ingresado fue el 23".

```
var evento = document.getElementById("evento");
evento.addEventListener('click',function(){
var entrada = document.querySelector("#entrada");
document.querySelector(".resultado").innerHTML = "El número ingresado
fue el " + entrada.value;
});
```

4. Desarrollar un programa en JavaScript que permita leer el nombre, correo electrónico y clave ingresados por el usuario en campos separados creados con elementos input. Cuando se active el evento del formulario mediante un click del usuario en un botón destinado para tal fin, debe aparecer un mensaje en una ventana emergente e indicar, por ejemplo: *"Hola Juan, tu correo electrónico es [usuario@usuario.com](#) y tu clave es 123456, bienvenido!!!"*. Puedes utilizar como base el código HTML del ejercicio anterior.

```
var formulario = document.getElementById("formulario");
formulario.addEventListener('submit',saludo);

function saludo(){
    var nombre = document.querySelector(".nombre");
    var correo = document.querySelector(".correo");
    var clave = document.querySelector(".clave");
    alert(`Hola ${nombre.value}, tu correo electrónico es:
${correo.value} y tu clave es ${clave.value}, Bienvenido`);
};
```

5. Al ejercicio propuesto número 4, agrégale la función preventDefault() para evitar el comportamiento por defecto del formulario.

```
var formulario = document.getElementById("formulario");
formulario.addEventListener('submit',saludo);

function saludo(event){
    event.preventDefault();
    var nombre = document.querySelector(".nombre");
    var correo = document.querySelector(".correo");
    var clave = document.querySelector(".clave");
    alert(`Hola ${nombre.value}, tu correo electrónico es:
${correo.value} y tu clave es ${clave.value}, Bienvenido`);
};
```


6. Un sitio web necesita implementar un pequeño formulario para registrar el correo de los usuarios que deseen recibir notificaciones. Para ello, debes validar el ingreso del dato en el campo del correo antes de poder procesar la información, es decir, el campo no puede estar sin datos, por esta razón, cuando el usuario haga un click sobre el botón el campo debe revisarse y verificar que contenga dato. Esta condición se debe mostrar en un mensaje en el caso de no existir dato alguno en el campo indicando que “debe existir un correo electrónico para poder registrarse”, de existir dato se debe mostrar, ejemplo: “tu correo electrónico usuario@correo.com fue registrado con éxito.” todo esto mediante un alert.

```
var formulario = document.getElementById("formulario");
formulario.addEventListener('submit',saludo);

function saludo(event){
    event.preventDefault();
    var correo = document.querySelector(".correo");
    if (correo.value != ''){
        alert(`Tu correo electrónico ${correo.value} fue registrado con
éxito`);
    }else{
        alert(`debe existir un correo electrónico para poder registrarse`);
    }
};
```

7. Para un formulario donde se solicita al usuario ingresar una palabra, la cual puede ser: “JavaScript, NodeJS o VueJS”. Crear una función que se active mediante un evento que permita validar que el texto ingresado por el usuario fue uno de los tres mencionados anteriormente. El mensaje debe ir en un alert e indicar: “Palabra permitida” o “Palabra no permitida”, dependiendo del caso.

```
var buscar = document.getElementById("buscar");
buscar.addEventListener('click',validar);

function validar(){
    var palabra = document.querySelector(".palabra").value;
    var patron1 = /JavaScript/;
    var patron2 = /NodeJS/;
    var patron3 = /VueJS/;
    if (palabra.match(patron1) || palabra.match(patron2) ||
palabra.match(patron3)){
        alert("Palabra permitida");
    } else {
```

```
    alert("Palabra no permitida");  
  };  
};
```

8. Para un formulario donde se solicita al usuario ingresar su edad, validar el ingreso de ese campo donde se permitirán números del 0 al 9. De lo contrario, no puede permitir el envío de la información, generando un mensaje de error en un alert. Para este ejercicio se cuenta con el extracto del formulario que tendrá disponible el usuario.

```
var enviar = document.getElementById("enviar");  
enviar.addEventListener('click', validar);  
  
function validar(){  
  var edad = document.querySelector(".edad").value;  
  var permitido = /[0-9]/gim;  
  if (edad.match(permitido)){  
    alert("El texto ingresado es permitido");  
  } else {  
    alert("El texto ingresado no es permitido");  
  };  
};
```

9. Partiendo del ejercicio guiado anterior (validando campos de un formulario), agrega ahora dos nuevos campos para ser validados. En este caso, debes agregar un campo para que el usuario pueda ingresar su apellido y otro campo para ingresar la edad, ambos del tipo texto.

HTML:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width,  
initial-scale=1.0">  
  <title>Document</title>  
</head>  
<body>  
  <form action="#" name="form" id="form">  
    <div>
```

```
        <label for="">Nombre</label>
        <input type="text" class="textNombre">
        <span class="errorNombre"></span>
    </div>
    <div>
        <label for="">Apellido</label>
        <input type="text" class="textApellido">
        <span class="errorApellido"></span>
    </div>
    <div>
        <label for="">Edad</label>
        <input type="text" class="textEdad">
        <span class="errorEdad"></span>
    </div>
    <div>
        <label for="">Teléfono</label>
        <input type="text" class="textTelefono">
        <span class="errorTelefono"></span>
    </div>
    <div>
        <label for="">Email</label>
        <input type="text" class="textEmail">
        <span class="errorEmail"></span>
    </div>
    <button type="submit">Enviar</button>
    <p class="resultado"></p>
</form>

    <script src="script.js"></script>
</body>
</html>
```

JavaScript:

```
let form = document.getElementById( "form" );

form.addEventListener( "submit", function ( event ) {
    event.preventDefault();
    limpiarErrores();
    let textNombre = document.querySelector(".textNombre").value;
    let textTelefono = document.querySelector(".textTelefono").value;
    let textEmail = document.querySelector(".textEmail").value;
    let textApellido = document.querySelector(".textApellido").value;
    let textEdad = document.querySelector(".textEdad").value;

    let resultado =
validar(textNombre,textTelefono,textEmail,textApellido, textEdad);

    if(resultado == true) {
        exito();
    };
});
function validar(nombre,telefono,email,apellido,edad) {
```